

DEPARTMENT OF COMPUTER SCIENCE

ADVANCED ARTIFICIAL INTILLIGENCE – CPS 581

Fall 2018

**Face Recognition using CNN**

**Professor:** Mehdi R. Zargham

**Team Members:**

Pranay Kumar Gade - (1016209350)

Somaraju Kamma - (101622121)

**Overview:**

* Introduction
* Network & utilities
* Training Dataset
* Testing Dataset
* Layers
* Code
* Accuracy
* Observations
* Future scope

**Introduction:**

Face recognition is of great importance to real world applications such as video surveillance, human machine interaction and security systems. As compared to traditional machine learning approaches, deep learning-based methods have shown better performances in terms of accuracy and speed of processing in image recognition.

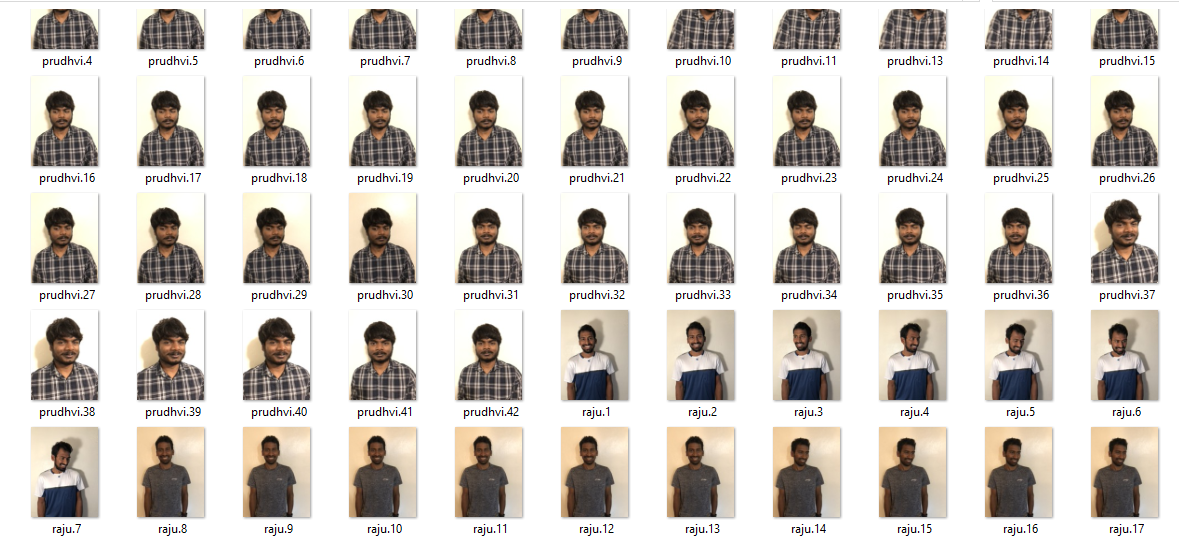
Simple Convolutional Neural Networks (CNN’s) work incredibly well at differentiating images, but can it work just as well at differentiating faces? Facial Recognition does of course use CNN’s in their algorithm, but they are much more complex, making them more effective at differentiating faces. In our project, we wanted to test how well a simple CNN with 5 convolutional layers would be able to distinguish faces.

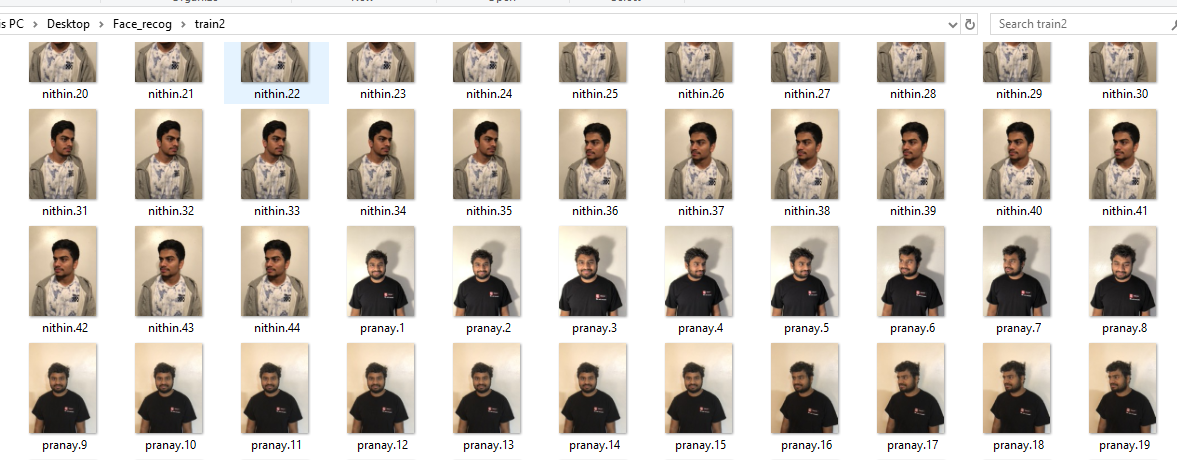
**Network & utilities:**

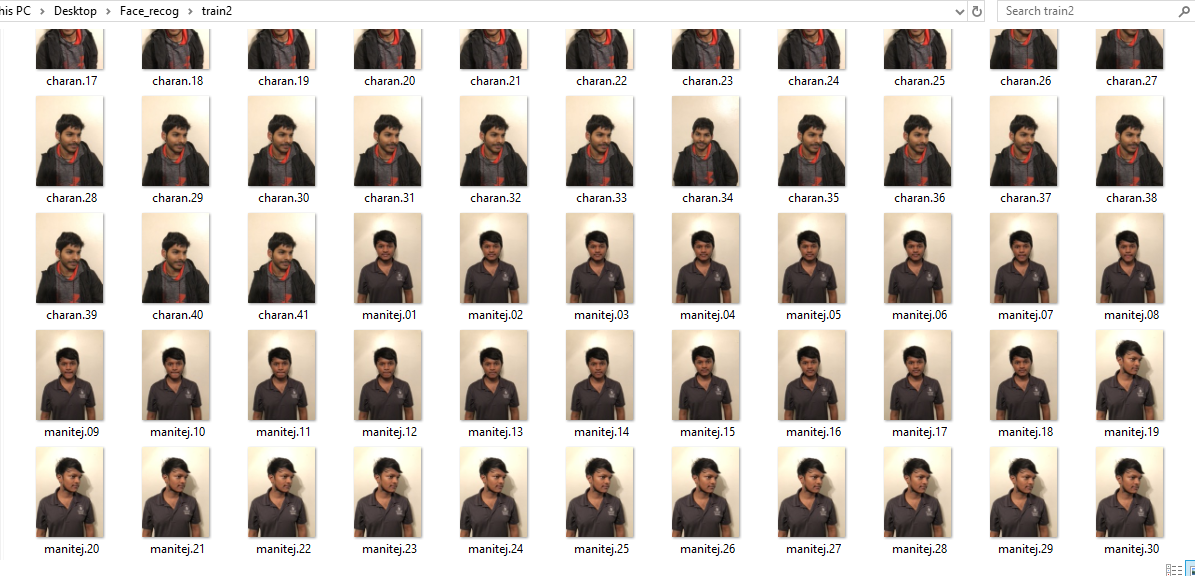
* CNN
* TensorFlow
* TFLearn
* Open CV for Image resizing and reshaping
* Numpy

**Training Dataset:**

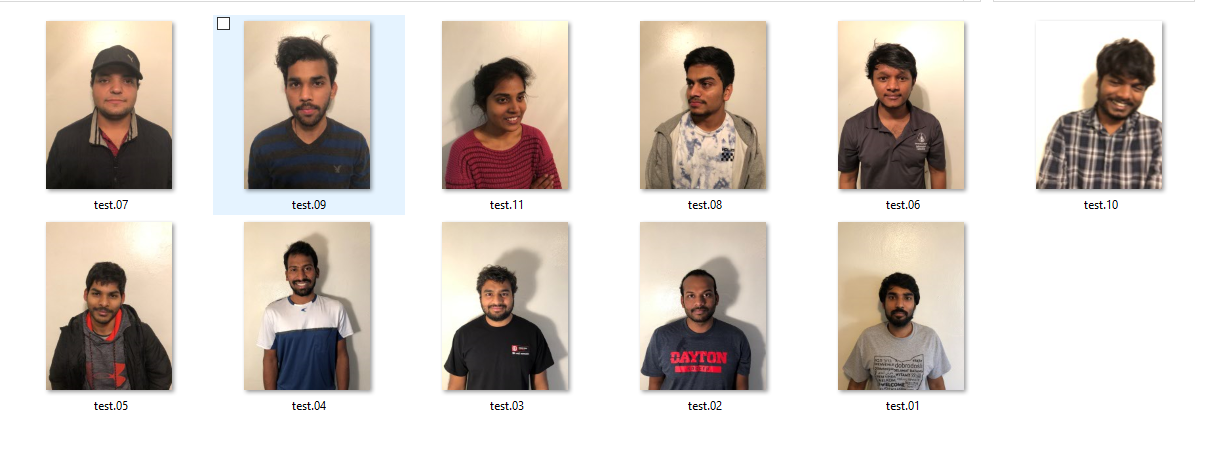
* We have considered 11 different people images, each person around 40-50 images.
* And each image is trained or identified using the label given to that image.





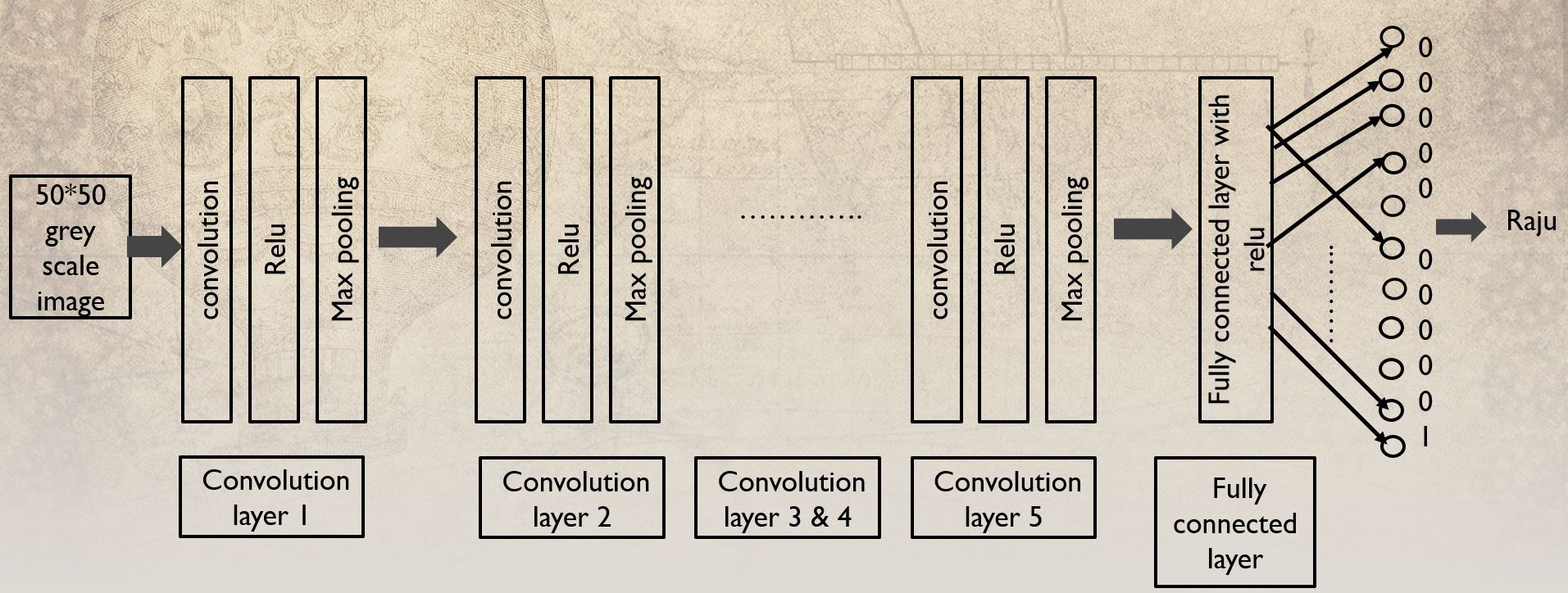


**Testing Dataset:**

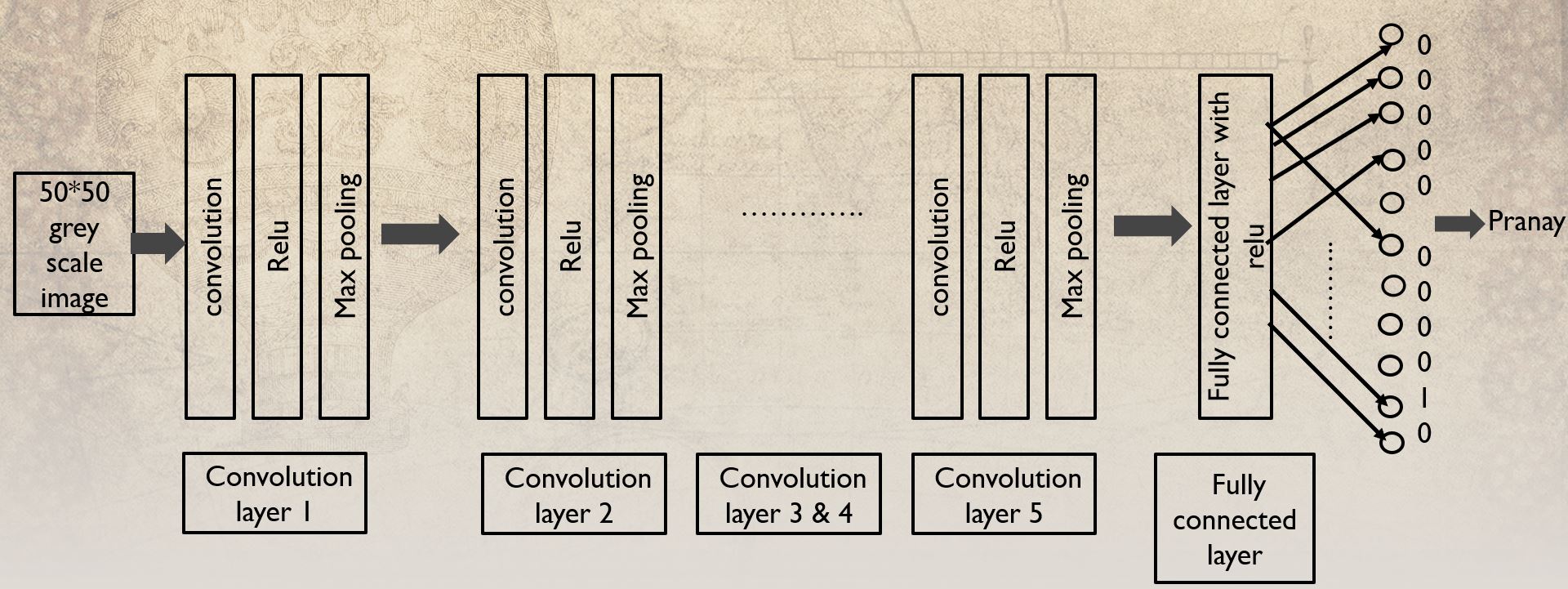


**Layers:**

For class1(“Raju”)



For class2(“Pranay”)



**Code:**

import cv2

import os

import numpy as np

from random import shuffle

from tqdm import tqdm

import tflearn

import tensorflow as tf

TRAIN\_DIR = 'C:/Users/pardhu/Desktop/Face\_recog/train2'

TEST\_DIR = 'C:/Users/pardhu/Desktop/Face\_recog/test2'

IMG\_SIZE = 50

LR = 1e-3

#LR = 0.0001

MODEL\_NAME = 'facerecog-{}-{}.model'.format(LR, '6conv-basic')

def label\_img(img):

word\_label = img.split('.')[-3]

#DIY One hot encoder

if word\_label == 'raju': return [0,0,0,0,0,0,0,0,0,0,1]

elif word\_label == 'pranay': return [0,0,0,0,0,0,0,0,0,1,0]

elif word\_label == 'sai': return [0,0,0,0,0,0,0,0,1,0,0]

elif word\_label == 'akhil': return [0,0,0,0,0,0,0,1,0,0,0]

elif word\_label == 'renuka': return [0,0,0,0,0,0,1,0,0,0,0]

elif word\_label == 'mohit': return [0,0,0,0,0,1,0,0,0,0,0]

elif word\_label == 'manitej': return [0,0,0,0,1,0,0,0,0,0,0]

elif word\_label == 'praneeth': return [0,0,0,1,0,0,0,0,0,0,0]

elif word\_label == 'nithin': return [0,0,1,0,0,0,0,0,0,0,0]

elif word\_label == 'charan': return [0,1,0,0,0,0,0,0,0,0,0]

elif word\_label == 'prudhvi': return [1,0,0,0,0,0,0,0,0,0,0]

def create\_train\_data():

training\_data = []

for img in tqdm(os.listdir(TRAIN\_DIR)):

label = label\_img(img)

path = os.path.join(TRAIN\_DIR,img)

img = cv2.imread(path,cv2.IMREAD\_GRAYSCALE)

img = cv2.resize(img, (IMG\_SIZE,IMG\_SIZE))

training\_data.append([np.array(img),np.array(label)])

shuffle(training\_data)

np.save('train\_data.npy', training\_data)

return training\_data

def process\_test\_data():

testing\_data = []

for img in tqdm(os.listdir(TEST\_DIR)):

path = os.path.join(TEST\_DIR,img)

img\_num = img.split('.')[0]

img = cv2.imread(path,cv2.IMREAD\_GRAYSCALE)

img = cv2.resize(img, (IMG\_SIZE,IMG\_SIZE))

testing\_data.append([np.array(img), img\_num])

shuffle(testing\_data)

np.save('test\_data.npy', testing\_data)

return testing\_data

#train\_data = create\_train\_data()

#test\_data = process\_test\_data()

train\_data = np.load('train\_data.npy')

test\_data = np.load('test\_data.npy')

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

tf.reset\_default\_graph()

convnet = input\_data(shape=[None, IMG\_SIZE, IMG\_SIZE, 1], name='input')

convnet = conv\_2d(convnet, 32, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 128, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 32, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = fully\_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)

convnet = fully\_connected(convnet, 11, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning\_rate=LR, loss='categorical\_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard\_dir='log')

if os.path.exists('C:/Users/pardhu/Desktop/Face\_recog/Cov\_Net{}.meta'.format(MODEL\_NAME)):

model.load(MODEL\_NAME)

print('model loaded!')

train = train\_data[:-244]

test = train\_data[-244:]

X = np.array([i[0] for i in train]).reshape(-1,IMG\_SIZE,IMG\_SIZE,1)

Y = [i[1] for i in train]

test\_x = np.array([i[0] for i in test]).reshape(-1,IMG\_SIZE,IMG\_SIZE,1)

test\_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n\_epoch=100, validation\_set=({'input': test\_x}, {'targets': test\_y}),

snapshot\_step=500, show\_metric=True, run\_id=MODEL\_NAME)

model.save(MODEL\_NAME)

import matplotlib.pyplot as plt

test\_data = np.load('test\_data.npy')

fig=plt.figure()

#predictions = classifier.predict(test\_data)

#print(convnet[0])

#print(\*convnet, sep = ", ")

#model\_out = model.predict([data])[0]

for num,data in enumerate(test\_data[:12]):

"""

if word\_label == 'raju': return [0,0,0,0,0,0,0,0,0,0,1]

elif word\_label == 'pranay': return [0,0,0,0,0,0,0,0,0,1,0]

elif word\_label == 'sai': return [0,0,0,0,0,0,0,0,1,0,0]

elif word\_label == 'akhil': return [0,0,0,0,0,0,0,1,0,0,0]

elif word\_label == 'renuka': return [0,0,0,0,0,0,1,0,0,0,0]

elif word\_label == 'mohit': return [0,0,0,0,0,1,0,0,0,0,0]

elif word\_label == 'manitej': return [0,0,0,0,1,0,0,0,0,0,0]

elif word\_label == 'praneeth': return [0,0,0,1,0,0,0,0,0,0,0]

elif word\_label == 'nithin': return [0,0,1,0,0,0,0,0,0,0,0]

elif word\_label == 'charan': return [0,1,0,0,0,0,0,0,0,0,0]

elif word\_label == 'prudhvi': return [1,0,0,0,0,0,0,0,0,0,0]

"""

img\_num = data[1]

img\_data = data[0]

y = fig.add\_subplot(4,5,num+1)

orig = img\_data

data = img\_data.reshape(IMG\_SIZE,IMG\_SIZE,1)

#model\_out = model.predict([data])[0]

model\_out = model.predict([data])[0]

print("output is :")

print(model\_out)

print(np.shape(model\_out))

print("1st element is :")

#print(model\_out[0,0])

model\_out1=np.array(model\_out).reshape(1, 11)

print("model out1[0,0] is :")

print(model\_out1[0,0])

if model\_out1[0,10] > 0.75 : str\_label='Raju'

elif model\_out1[0,9] > 0.75 : str\_label='Pranay'

elif model\_out1[0,8] > 0.75 : str\_label='Sai'

elif model\_out1[0,7] > 0.75 : str\_label='Akhil'

elif model\_out1[0,6] > 0.75 : str\_label='Renuka'

elif model\_out1[0,5] > 0.75 : str\_label='Mohit'

elif model\_out1[0,4] > 0.75 : str\_label='Manitej'

elif model\_out1[0,3] > 0.75 : str\_label='Praneeth'

elif model\_out1[0,2] > 0.75 : str\_label='Nithin'

elif model\_out1[0,1] > 0.75 : str\_label='Charan'

elif model\_out1[0,0] > 0.75 : str\_label='Prudhvi'

else: str\_label='Unknown'

#if np.argmax(model\_out) == 0 : str\_label='Raju111'

#else: str\_label='Pranay'

y.imshow(orig,cmap='gray')

plt.title(str\_label)

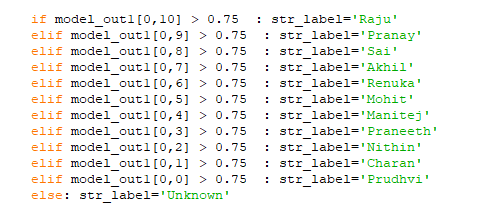
y.axes.get\_xaxis().set\_visible(False)

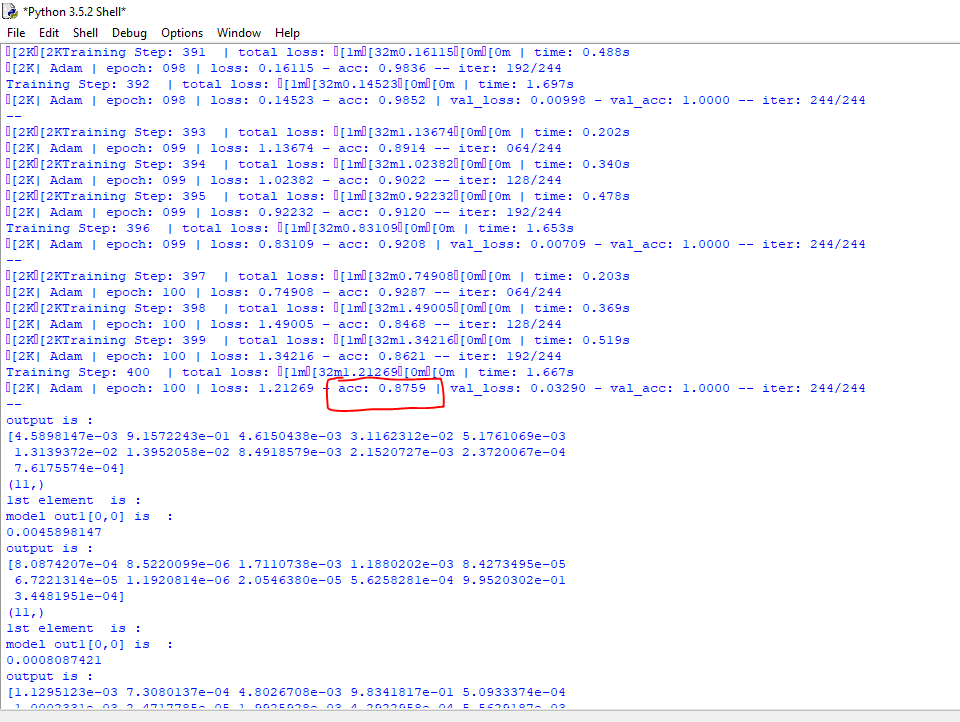
y.axes.get\_yaxis().set\_visible(False)

plt.show()

**Accuracy:**

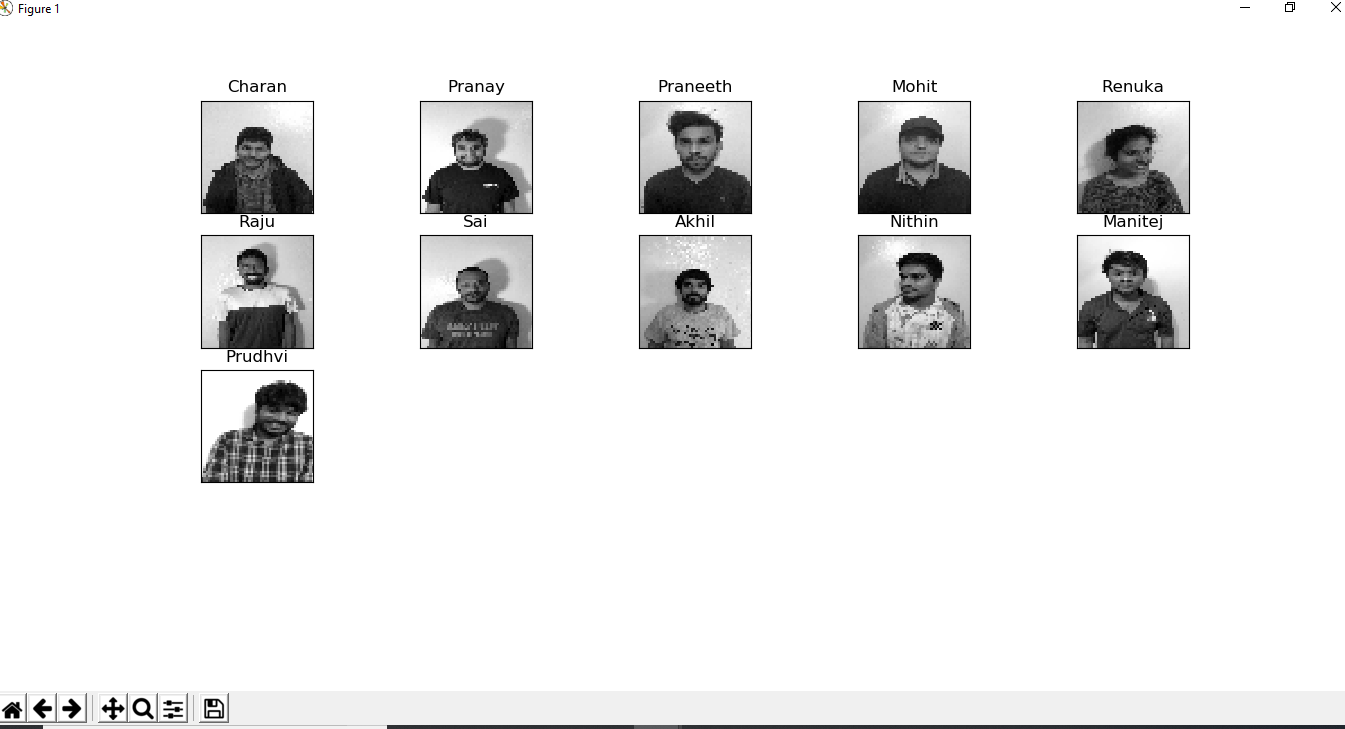
* We have defined for more than 75% of features of a single person to recognize his identity.
* We are getting around 87 to 92% accuracy.





**Observations:**

We have got the following results, which are accurate.



**Future scope:**

* We want to increase the scope of the project by increasing number of training Dataset (Classes)
* We want to achieve 99.99% accuracy
* We also want to train the data with different background
* We want to test with Twins images (which are similar)
* Moreover, we want to implement age detection.